Algoritma Analizi

1

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

Algoritma Analizi

Her hangi bir programlama dilinde yazılmış bir algoritmanın ne kadar hızlı çalıştığını veya ne kadar sürede çalıştığını o algoritmayı analiz ederek yapabiliriz. Peki, algoritma analizi nedir? Algoritma analizi denince akla iki önemli kavram gelir bunlar alan ve zaman karmaşıklığıdır.

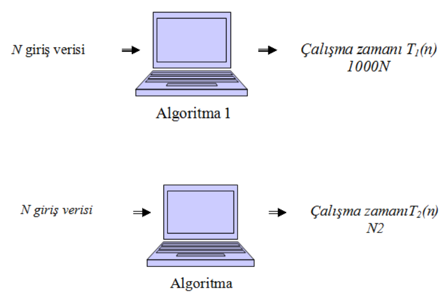
Alan karmaşıklığı yazdığınız algoritma bellekten ne kadar yer kullanıyor, zaman karmaşıklığı ise yazdığınız algoritmanın çalışma süresini ifade eder. Algoritma analizine neden ihtiyaç duyarız çünkü yazdığımız algoritmanın performansını bilmek isteriz, farklı algoritmalarla karşılaştırmak isteriz ve daha iyisi mümkün mü sorusuna ancak analiz yaparak cevap verebiliriz.

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

2

ALGORİTMA ANALİZİ

Aynı işi yapan fakat farklı yazılmış iki algoritmaya yani verileri girdi olarak verdiğimizde farklı çalışma zamanlarında işlerini bitirdiklerini gözleriz.



Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

ALGORİTMA ANALİZİ

Bir algoritma çalışmasını bitirene kadar geçen süre yürütme zamanı olarak adlandırılır. Ve algoritmada genelde eleman sayısı n olarak gösterilir ve yürütme zamanı da T(n) ile ifade edilir. Algoritmadaki eleman sayısı çok fazla olduğunda yürütme zamanı, zaman karmaşıklığı olarak adlandırılır. Ve derecesi asimptotik notasyon ile verilir. O(o), Θ(o) veya Ω(o) gibi notasyonlar kullanılmaktadır.

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

ALGORİTMA ANALİZİ

Algoritmanın işlevini yerine getirmesi için kullandığı bellek miktarına alan maliyeti denir. Örneğin n elemanlı bir dizi, elemanlarının her biri 4 byte ise bu dizi için alan maliyeti bağıntısı;

$$T(n) = 4n$$

Aynı şekilde alan karmaşıklığı da eleman sayısı çok büyük olduğu zaman alan maliyetini ifade eden asimptotik ifadedir. Zaman karmaşıklığında kullanılan notasyonlar burada da kullanılır.

Dr. Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

ALGORİTMA ANALİZİ

Basitçe bir algoritmanın zaman karmaşıklığını hesaplamak o algoritmada operasyon sayısını saymaktır. Bu zaman karmaşıklığı derleyiciden bağımsız olmalıdır. Ve yine hesap yaparken bir çok ayrıntıyı da göz ardı etmemiz gerekir.

Big Oh Notasyonu O(n)

Paul Bachman tarafından tanıtılmıştır. Zaman karmaşıklığında üst sınırı gösterir. Bu notasyon bir çok ifadeyi sadeleştirerek göstermemizi sağlar. Örneğin n^3 + n^2+3n gibi bir ifadeyi O(n^3) olarak ifade ederiz.

Big Omega Notasyonu

Big Oh notasyonunun tam tersidir. Zaman karmaşıklında alt sınırı gösterir. Yani Omega ile ölçülen değerden daha hızlı bir değer elde etmeniz mümkün değildir.

Dr. Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

ALGORİTMA ANALİZİ

Big Theta Notasyonu

Bu notasyon big Oh notasyonu ile big Omega notasyonu arasında ortalama bir karmaşıklığı ifade eder.

Algoritma analizinde en iyi, ortalama ve en kötü durum nedir? Bunlara kısaca değinecek olursak en iyi durum lower bound'u ifade eder yani bu ifadeden daha hızlı algoritma çalıştırılamaz. Ortalama zaten upper ve lower bound'un ortalama bir değer ifade eder. Upper bound ise bir algoritma upper bounddan daha yavaş çalıştırılamaz. Aşağıda örnek bir algoritmanın analizini yapalım.

Sum = 0; ------------------------ 1 kez çalıştırılır

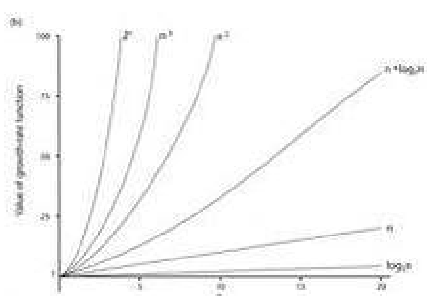
For (int i = 0; i < N; i++) ----------------------- n+1 kez çalıştırılır

Sum = Sum + X[i]; -------------------- n kez çalıştırılır

Algoritmanın zaman karmaşıklığı T(n) = 1 + n +1 + n = 2n+2

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

7



Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

8

ÖRNEK I: DİZİDEKİ SAYILARIN TOPLAMINI BULMA

```
int Topla(int A[], int N)
{
  int toplam = 0;

  for  (i=0; i < N; i++){
    toplam += A[i];
  } //Bitti-for

  return toplam;
} //Bitti-Topla
```

- Bu fonksiyonun yürütme zamanı ne kadardır?

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

9

ÖRNEK I: DİZİDEKİ SAYILARIN TOPLAMINI BULMA

```
int Topla(int A[], int N)
{
  int topla = 0;              ---> 1

  for (i=0; i < N; i++){      ---> N+1
    topla += A[i];            ---> N
  } //Bitti-for

  return topla;               ---> 1
} //Bitti-Topla
```

İşlem sayısı: 1, N+1, N, 1

Toplam: 1 + N + N +1+ 1 = 2N + 3

- Çalışma zamanı: T(N) = 2N+3
  - N dizideki sayı sayısı

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

10

ÖRNEK II: DİZİDEKİ BİR ELEMANIN ARANMASI

```
int Arama(int A[], int N,
                  int sayi) {
  int i = 0; ---------------------------------------->  1

  while (i < N){ ------------------------------------> 1<=L<=N+1
    if (A[i] == sayi) break;-------------------------> 1<=L<=N
    i++;---------------------------------------------> 0<=L<=N
  } //bitti-while

  if (i < N) return i;-------------------------------> 1
  else return -1;------------------------------------> 1
} //bitti-Arama
```

İşlem sayısı: 1, 1<=L<=N+1, 1<=L<=N, 0<=L<=N, 1, 1

Toplam: 1+3*L+1+1+1 = 3L+4

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

ÖRNEK II: DİZİDEKİ BİR ELEMANIN ARANMASI

- En iyi çalışma zamanı nedir?
  - Döngü sadece bir kez çalıştı=>T(n) = 7
- Ortalama(beklenen) çalışma zamanı nedir?
  - Döngü N/2 kez çalıştı =>T(n)=3*n/2+3 = 1.5n+4
- En kötü çalışma zamanı nedir?
  - Döngü N kez çalıştı =>T(n) = 3n+4

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

ALGORİTMALARIN EN KÖTÜ DURUM ANALİZİ

- Bir algoritmanın sadece EN KÖTÜ durumdaki çalışma zamanına bakılır. Neden?
  - En kötü durum çalışma zamanında bir üst sınırdır ve o algoritma için verilen durumdan daha uzun sürmeyeceği garantisi verir.
  - Bazı algoritmalar için en kötü durum oldukça sık rastlanır. Arama algoritmasında, aranan öğe genellikle dizide olmaz dolayısıyla döngü N kez çalışır.
  - Ortalama çalışma zamanı genellikle en kötü çalışma zamanı kadardır. Arama algoritması için hem ortalama hem de en kötü çalışma zamanı doğrusal fonksiyondur.

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

ÖRNEK III: İÇ İÇE DÖNGÜLER

```
for (i=1; i<=N; i++){
    for (j=1; j<=N; j++){
        printf("Foo\n");
    } //bitti-içteki for
} //bitti-dıştaki for
```

- Prinf fonksiyonu kaç kez çalıştırıldı?
  - Veya Foo yazısı ekrana kaç kez yazılır?

$$T(N) = \sum_{i=1}^{N} \sum_{j=1}^{N} 1 = \sum_{i=1}^{N} N = (N+1)*(N+1) = N^2 + 2N + 1$$

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

Örnek IV: Matris Çarpımı

```
/* İki boyutlu dizi A, B, C.      Hesapla C = A*B    */
for (i=0; i<N; i++) {
    for (j=0; j<N; j++) {
        C[i][j] = 0;
        for (int k=0; k<N; k++){
          C[i][j] += A[i][k]*B[k][j];
        } //bitti-en içteki for
    } //bitti-içteki for
} //bitti-dıştaki for
```

$$T(N) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left(1 + \sum_{k=0}^{N-1} 1\right) = N^3 + N^2$$

Dr. Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

Örnek V: İkili Arama

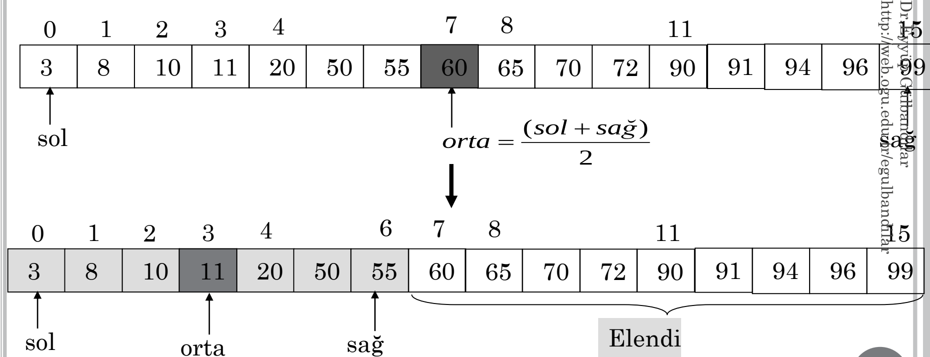
- Problem: Sıralı bir dizi veriliyor ve bir sayıyı arıyorsunuz.
  - Doğrusal arama– T(n) = 3n+2 (En kötü durum)
  - Daha iyisi yapılabilir mi?
  - Ö.g. Aşağıdaki sıralı dizide 55 sayısını arayalım

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	8	10	11	20	50	55	60	65	70	72	90	91	94	96	99

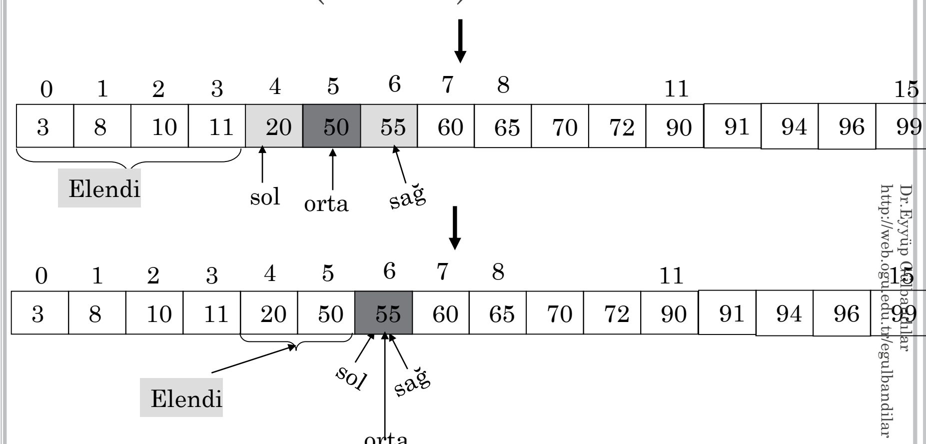
Dr. Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

Örnek V: İkili Arama

- Dizi sıralanmış olduğundan, dizinin ortasında bulunan sayı ile aranan sayıyı karşılaştırarak arama boyutunu yarıya düşürülür ve bu şekilde devam edilir.
- Örnek: 55'i arayalım

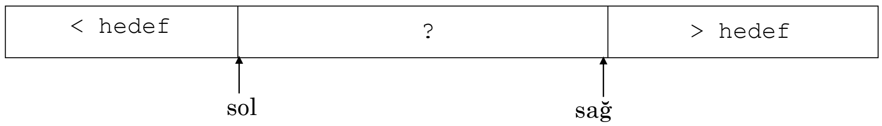


İkili arama (devam)



- 55'i bulduk→ Başarılı arama
- 57'yi aradığımızda, bir sonraki işlemde başarısız bir şekilde sonlanacak.

İKİLİ ARAMA (DEVAM)



- Hedefi ararken herhangi bir aşamada, arama alanımızı "sağ" ile "sol" arasındaki alana kısıtlamış oluyoruz.
- "sol" 'un solunda kalan alan hedeften küçüktür ve bu alan arama alanından çıkarılır.
- "sağ" ın sagında kalan alan hedeften büyüktür ve bu alan arama alanından çıkarılır.

İKİLİ ARAMA - ALGORİTMA

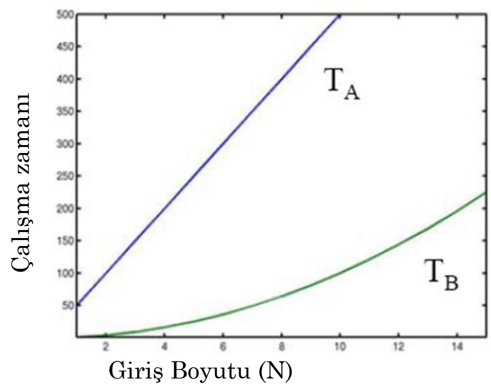
```
// Aranan sayının indeksini döndürür aranan sayı bulunamazsa -1 döndürür.
int ikiliArama(int A[], int N, int sayi){
    sol = 0;
    sag = N-1;

    while (sol <= sag){
        int orta = (sol+sag)/2;              // Test edilecek sayının indeksi
        if (A[orta] == sayi) return orta;   // Aranan sayı bulundu. İndeksi döndür
        else if (sayi < A[orta]) sag = orta – 1;      // Sağ tarafı ele
        else sol = orta+1;                            // Sol tarafı ele
    } //bitti-while

    return –1;   // Aranan sayı bulunamadı
} //bitti-ikiliArama
```

- En kötü çalışma zamanı: $T(n) = 3 + 5*\log_2 N$. Neden?

ASİMPTOTİK NOTASYON

- Bir problemi çözmek için A ve B şeklinde iki algoritma verildiğini düşünelim.
- Giriş boyutu N için aşağıda A ve B algoritmalarının çalışma zamanı T_A ve T_B fonksiyonları verilmiştir.

- Hangi algoritmayı seçersiniz?

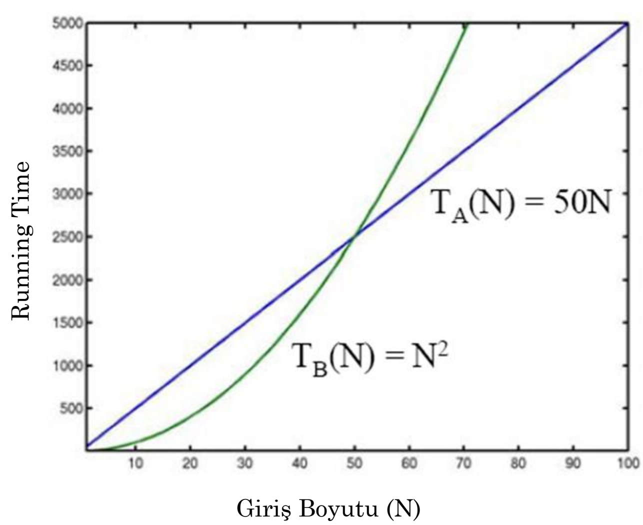


Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

21

ASİMPTOTİK NOTASYON (DEVAM)

- N büyüdüğü zaman A ve B nin çalışma zamanı:



- Şimdi hangi algoritmayı seçersiniz?

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

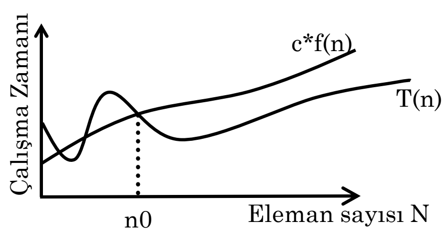
22

Asimptotik Notasyon (devam)

- Genel olarak, asimptotik notasyon, eleman sayısı n'nin sonsuza gitmesi durumunda algoritmanın, benzer işi yapan algoritmalarla karşılaştırmak için kullanılır.
- Eleman sayısının küçük olduğu durumlar pratikte mümkün olabilir fakat bu birçok uygulama için geçerli değildir.
- Verilen iki algoritmanın çalışma zamanını T1(N) ve T2(N) fonksiyonları şeklinde gösteriyoruz. Fakat hangisinin daha iyi olduğunu belirlemek için bir yol belirlememiz gerekiyor. (asimptotik olarak daha küçük gibi)
  - Asimptotik notasyonlar
  - Büyük-Oh, Ω, Θ notasyonları

Büyük-Oh(Big-Oh) Notasyonu: Asimptotik Üst Sınır

- T(n) = O(f(n))
  - c ve n_0 şeklinde pozitif sabitlerimiz olduğunu düşünelim. n >= n_0 ifadesini sağlayan tüm değerler için T(n) <= c*f(n) dir.



- Örnek: T(n) = 50n → O(n). Neden?
  - c=50, n_0=1 seçersek. n>=1 için 50n <= 50n olur.
  - Başka uyan sayılarda mevcuttur.

Büyük-Oh(Big-Oh) Notasyonu: Asimptotik Üst Sınır

- $T(n) = O(f(n))$
  - c ve n_0 şeklinde pozitif sabitlerimiz olduğunu düşünelim. $n >= n_0$ ifadesini sağlayan tüm değerler için $T(n) <= c*f(n)$ dir.
- Örnek: $T(n) = 2n+5$ is $O(n)$ Neden?
  - $n>=n_0$ şartını sağlayan tüm sayılar için $T(n) = 2n+5 <= c*n$
  - $n>=1$ için $2n+5 <= 2n+5n <= 7n$
    - $c = 7$, $n_0 = 1$
  - $n>=5$ şartını sağlayan tüm sayılar için $2n+5 <= 3n$
    - $c = 3$, $n_0=5$
  - Diğer c ve n_0 değerleri de bulunabilir.

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

25

Büyük-Oh(Big-Oh) Notasyonu: Asimptotik Üst Sınır

- $T(n) = O(f(n))$
  - c ve n_0 şeklinde pozitif sabitlerimiz olduğunu düşünelim. $n >= n_0$ ifadesini sağlayan tüm değerler için $T(n) <= c*f(n)$ dir.
- Örnek: $T(n) = 2n+5$ is $O(n^2)$ Neden?
  - $n>=n_0$ şartını sağlayan tüm sayılar için $T(n) = 2n+5 <= c*n^2$ şartını sağlayan c ve n0 değerlerini arıyoruz.
  - $n>=4$ için $2n+5 <= 1*n^2$
    - c = 1, no = 4
  - $n>=3$ için $2n+5 <= 2*n^2$
    - c = 2, no = 3
  - Diğer c ve n_0 değerleri de bulunabilir.

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

26

Büyük-Oh(Big-Oh) Notasyonu: Asimptotik Üst Sınır

- T(n) = O(f(n))
  - c ve n_0 şeklinde pozitif sabitlerimiz olduğunu düşünelim. n >= n_0 ifadesini sağlayan tüm değerler için T(n) <= c*f(n) dir.
- Örnek: T(n) = n(n+1)/2 → O(?)
  - T(n) = n^2/2 + n/2 → O(N^2). Neden?
  - n >= 1 iken $n^2/2 + n/2 <= n^2/2 + n^2/2 <= n^2$
  - Böylece, T(n)=n*(n+1)/2 <= 1* n^2 for all n >=1
    - c=1, no=1
  - Not: T(n) ayrıca O(n^3) tür.

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

27

Karşılaşılan Genel Fonksiyonlar

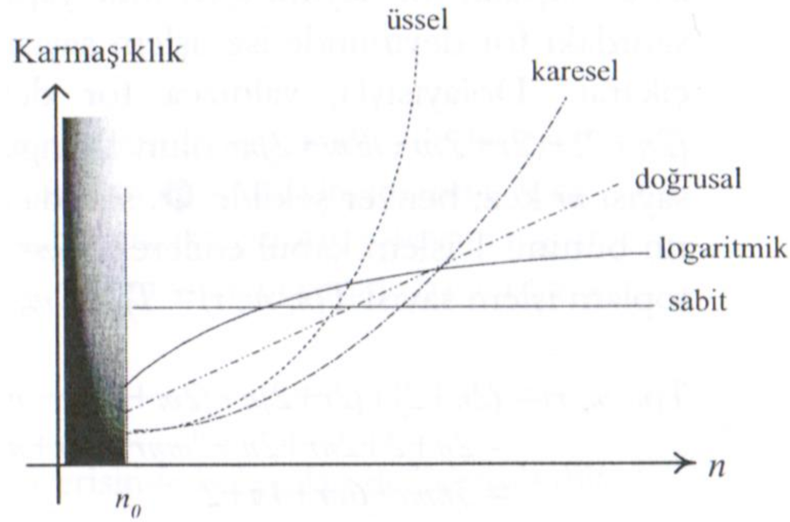
Maliyet artar

İsim	Büyük-Oh	Yorum
Sabit	O(1)	Yenilmez!
Log log	O(loglogN)	Tahminsel arama
Logaritmik	O(logN)	İyi hazırlanmış arama algoritmalarının tipik zamanı
Doğrusal	O(N)	Hızlı bir algoritmadır. N tane veriyi girmek için gereken zaman.
N logN	O(NlogN)	Çoğu sıralama algoritması
Karesel	O(N^2)	Veri miktarı az olduğu zamanlarda uygun (N<1000)
Kübik	O(N^3)	Veri miktarı az olduğu zamanlarda uygun (N<1000)
Üssel	O(2^N)	Veri miktarı çok az olduğunda uygun (n<=20)

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

28

KARŞILAŞILAN GENEL FONKSİYONLAR (DEVAM)



Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

29

ÖRNEK: MAKSİMUM ALT DİZİ TOPLAMI

- Tanım: Verilen bir tamsayı listesi içerisinde/dizisinde ***elemanları komşu olmak şartıyla*** hangi (bitişik) alt dizi en yüksek toplamı verir?
- Örneğin:
  - { -2,11,-4,13,-5,2 } → Cevap=20
  - { 1,2,-5,4,7,-2 } → Cevap=11
  - { 1,5,-3,4,-2,1 } → Cevap=7
- Bu problemi çözen çok sayıda algoritma vardır.

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

30

Çözüm-1 Kaba Kuvvet Algoritması

```
public static int maxAltDiziT( int[] a){
    int maxTop = 0;
    for(int i=0; i<a.length; i++)
        for(int j=i; j<a.length; j++){
            int top=0;
            for(int k=i; k<=j; k++)
                top += a[k];
            if(top > maxTop){
                maxTop = top;
                int bas = i;    // alt dizinin başlangıcı
                int son = j;    // alt dizinin sonu
            }
        }
    return maxTop;
}
```

Bu algoritmanın karmaşıklığı nedir?

$2n^3+6n^2+n+2 \rightarrow O(n^3)$

Daha iyisi yapılabilir mi?

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

31

Çözüm-2 Geliştirilmiş Algoritma

```
public static int maxAltDiziT(int[] a) {
    int maxTop = 0;
    for (int i = 0; i < a.length; i++) {
        int top = 0;
        for (int j = i; j <= a.length; j++) {
            top += a[j];
            if (top > maxTop) {
                maxTop = top;
                int bas = i;    // alt dizinin başlangıcı
                int son = j;    // alt dizinin sonu
            }
        }
    }
    return maxTop;
}
```

Bu algoritmanın karmaşıklığı nedir?

$6n^2+2n+2 \rightarrow O(n^2)$

Daha iyisi yapılabilir mi?

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

32

Çözüm-3 Doğrusal Algoritma

```
public static int maxAltDiziT(int[] a) {
    int maxTop = 0;
    int top = 0;
    for (int i=0, j=0; j<=a.length; j++) {
        top += a[j];
        if (top > maxTop) {
            maxTop = top;
            int bas = i;    // alt dizinin başlangıcı
            int son = j;    // alt dizinin sonu
        } else if (top<0){
            i = j + 1;
            top = 0;
        }
    }
    return maxTop;
}
```

Bu algoritmanın karmaşıklığı nedir?

9n+3 → O(n)

Daha iyisi yapılabilir mi?

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

33

Maksimum Alt Dizi Toplamı Çalışma Süresi

Çeşitli *Maksimum Alt Dizi Toplamı* algoritmaları için çalışma süreleri aşağıda verilmiştir. (saniye cinsinden)

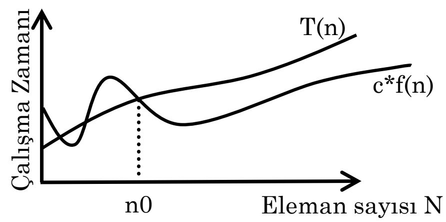
N	$O(N^3)$	$O(N^2)$	O(N log N)	O(N)
10	0,000001	0,000000	0,000001	0,000000
100	0,000288	0,000019	0,000014	0,000005
1 000	0,223111	0,001630	0,000154	0,000053
10 000	218	0,133064	0,001630	0,000533
100 000	NA	13,17	0,017467	0,005571
1 000 000	NA	NA	0,185363	0,056338

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

34

Ω Notasyonu: Asimptotik Alt Sınır

- $T(n) = \Omega(f(n))$
  - c ve n_0 şeklinde pozitif sabitlerimiz olduğunu düşünelim. $n >= n_0$ ifadesini sağlayan tüm değerler için $T(n) >= c*f(n)$ dir.



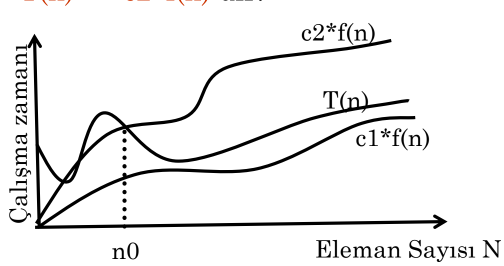
- Örnek: T(n) = 2n + 5 → Ω(n). Neden?
  - 2n+5 >= 2n, tüm n >= 1 için
- T(n) = 5*n² - 3*n → Ω(n²). Neden?
  - 5*n² - 3*n >= 4*n², tüm n >= 4 için

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

35

Θ Notasyonu: Asimptotik Alt ve Üst Sınır

- $T(n) = \Theta(f(n))$
  - c1,c2 ve n_0 şeklinde pozitif sabitlerimiz olduğunu düşünelim $n >= n_0$ ifadesini sağlayan tüm değerler için $c1*f(n) <= T(n) <= c2*f(n)$ dir.



- Örnek: T(n) = 2n + 5 → Θ(n). Neden?

  2n <= 2n+5 <= 3n, tüm n >= 5 için
- T(n) = 5*n² - 3*n → Θ(n²). Neden?
  - 4*n² <= 5*n² - 3*n <= 5*n², tüm n >= 4 için

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

36

Büyük-Oh, Theta, Omega

İpucu:

- O(f(N)) düşünürsek f(N) ile "eşit veya küçük"
  - Üstten sınır: f(N) ile "yavaş veya aynı hızda büyür"
- Ω(f(N)) düşünürsek f(N) ile "eşit veya büyük"
  - Alttan sınır: f(N) ile "aynı hızda veya hızlı büyür"
- Θ(f(N)) düşünürsek f(N) ile "eşit"
  - Alttan ve Üsten sınır : büyüme oranları eşit
- (*N'nin büyük olduğu ve sabiterin elendiği durumlarda*)

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

37

Sıkça Yapılan Hatalar

- Karmaşıklığı bulmak için sadece döngüleri saymakla yetinmeyin.
  - 2 içi içe döngünün 1 den N^2 kadar döndüğünü düşünürsek karmaşıklık $O(N^4)$ olur.
- $O(2N^2)$ veya $O(N^2+N)$ gibi ifadeler kullanmayın.
  - Sadece baskın terim kullanılır.
  - Öndeki sabitler kaldırılır.
- İç içe döngüler karmaşıklığı direk etkilerken art arda gelen döngüler karmaşıklığı etkilemez.

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

38

Bazı Matematiksel İfadeler

S(N) = 1 + 2 + 3 + 4 + … N = $\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$

Karelerin Toplamı: $\sum_{i=1}^{N} i^2 = \frac{N*(N+1)*(2n+1)}{6} \approx \frac{N^3}{3}$

Geometrik Seriler: $\sum_{i=0}^{N} A^i = \frac{A^{N+1}-1}{A-1}$ A > 1

A < 1

$$\sum_{i=0}^{N} A^i = \frac{1-A^{N+1}}{1-A} = \Theta(1)$$

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

39

Bazı Matematiksel İfadeler

Lineer Geometrik seriler: $\sum_{i=0}^{n} ix^i = x + 2x^2 + 3x^3 + \ldots + nx^n = \frac{(n-1)x^{(n+1)} - nx^n + x}{(x-1)^2}$

Harmonik seriler: $H_n = \sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} = (\ln n) + O(1)$

Logaritma:

$$\log A^B = B * \log A$$

$$\log(A * B) = \log A + \log B$$

$$\log(\frac{A}{B}) = \log A - \log B$$

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar

40

Bazı Matematiksel İfadeler

- İki sınır arasındaki sayıların toplamı:

$$\sum_{i=a}^{b} f(i) = \sum_{i=0}^{b} f(i) - \sum_{i=0}^{a-1} f(i)$$

$$\sum_{i=1}^{n} (4i^2 - 6i) = 4\sum_{i=1}^{n} i^2 - 6\sum_{i=1}^{n} i$$

Dr.Eyyüp Gülbandılar
http://web.ogu.edu.tr/egulbandilar